

Write-up for the Robo-ND deep learning project

Goal

In this project I implemented a deep-learning network for semantic segmentation at pixel level for identifying a target person so a quadrotor can follow it in the simulator.

Model description and training process

My network was an FCN network with three encoder blocks with depth 32, 64, 128. After the encoders I used a 1x1 convolution layer with 256 depth and then three decoders which each one of them does upsampling by a number of two. A final convolution layer was used for classification. I trained the model with the data provided by Udacity and I got an accuracy of 45%. Then I tested the model on the simulator and the quadrotor started following the target as expected.

Model architecture details

Encoders

The FCN architecture is commonly used for image classification and especially for identifying every pixel in the image. The encoder layers are used for identifying useful information from the input image. The first encoder layer is trained to identify simple features and the next encoder layers identify more and more complicated features and shapes.

1x1 convolution

The 1x1 convolutional layer is used when we want to flatten a convolutional layer without loss of spatial information.

Decoders

The decoder layers are used to upscale the output of the encoder such as it's the same size of the original image. Therefore it allows prediction of each pixel in the original image.

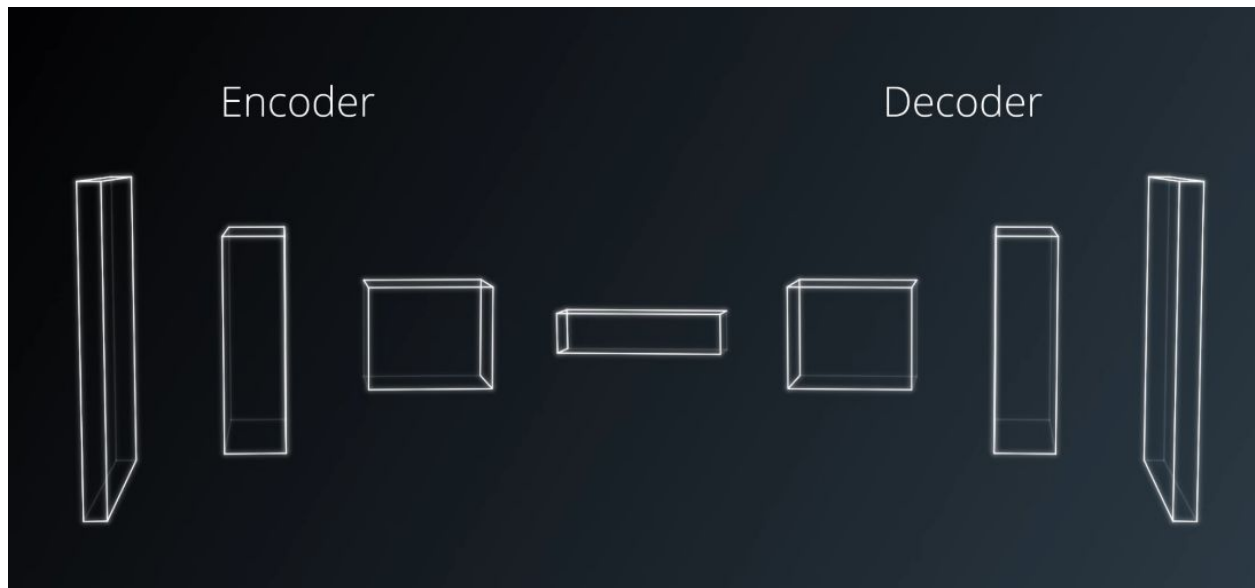
Skip connections

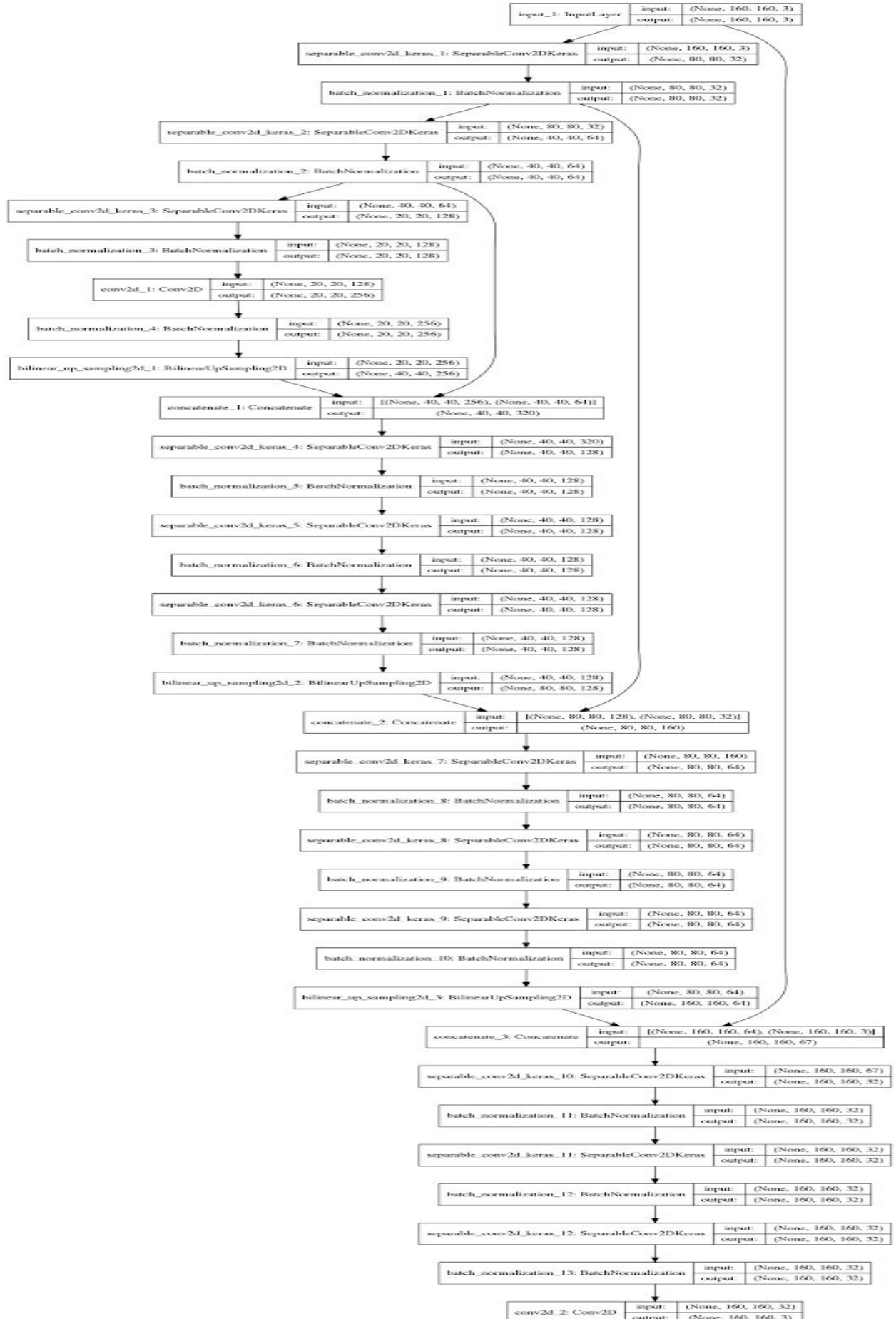
Also there are 3 skip connections as shown in the diagram below. These skip connections are used to retain the information that would be lost from the encoding to decoding process and allow the network to use information from multiple resolution scales.

Final convolutional layer

The last convolutional layer of the model is used to make the classification of each pixel using the softmax activation function.

Network model images





Tensorflow and keras implementation

Each encoder block consists of a separable convolution with batch normalization. This was implemented in the `encoder_block` function using:

```
output_layer = separable_conv2d_batchnorm(input_layer, filters, strides)
```

The decoder block was implemented in the `decoder_block` function and consisted of :

Upsample the small input layer using `bilinear_upsample()` and the concatenate it with the large input layer.

After that I added three separable convolution layers with batch normalization. I found that 2 to 3 convolution layer work well on training the network.

The FCN model was created inside the `fcn_model` function.

It consists of 3 encoder blocks that each block increase the depth , then a 1x1 convolution layer with batch normalization followed by three decoder blocks.

The final layer is a 2D convolution layer with softmax activation function and 'same' padding.

Model tries and hyperparameters

I tried different FCN models for training but I observe that using more than three encoder blocks the model was overfitting. With fewer blocks I got an accuracy of 32%.

While training the network I used for 5 to 20 epochs using my GPU GTX960M. I didn't use more than 20 epochs because the training time was too big and I didn't get any additional benefit.

I tried learning rate of 0.1, 0.01 and 0.001 and I got the best results with 0.01.

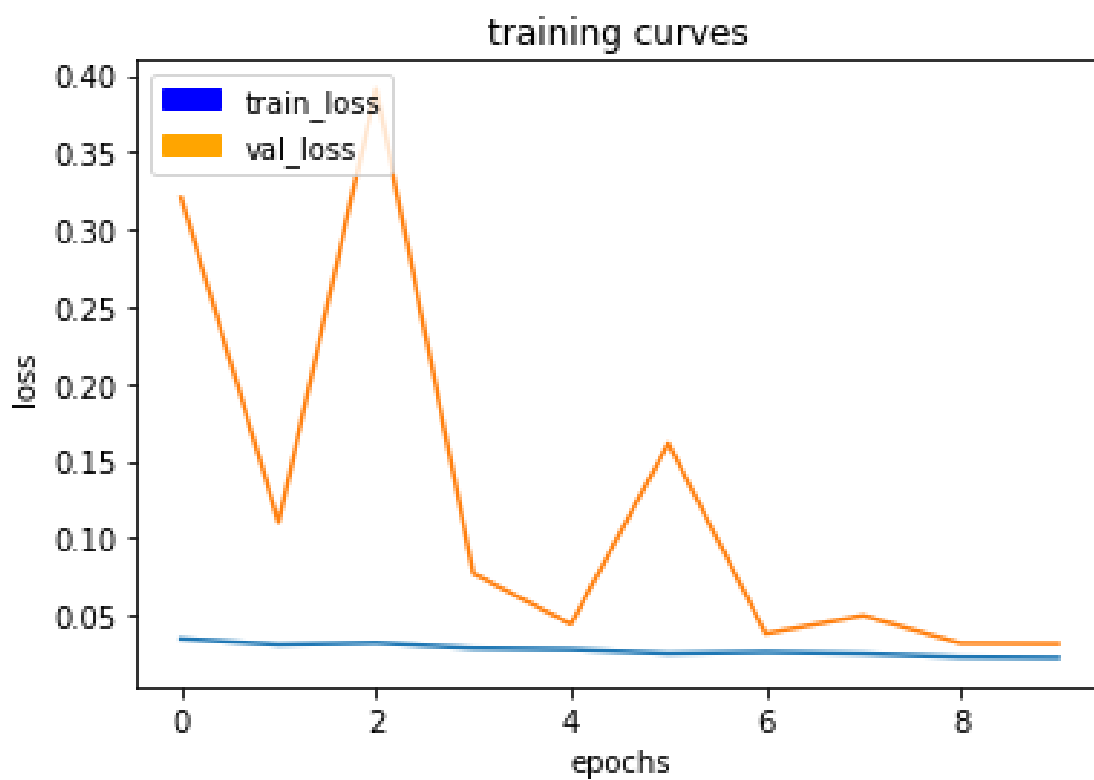
Batch size was set to 32 and `num_epochs` = 10.

I left `steps_per_epoch` = 100 and `validation_steps` = 50 constant throughout my different training tries for computation issues.

For workers > 2 my GPU was crushing so I kept it at two.

Training and results

The loss graph of my final model that achieved 45% accuracy is shown below. I was afraid that it was overfitted because of the validation loss sudden increases but the model seems to work fine.



My network produces the images on the right part:

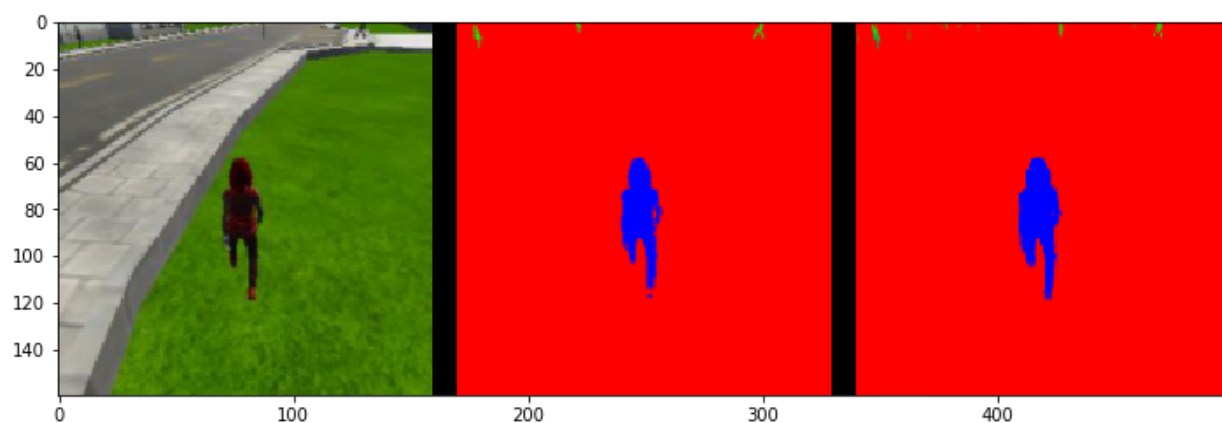


Image with target following

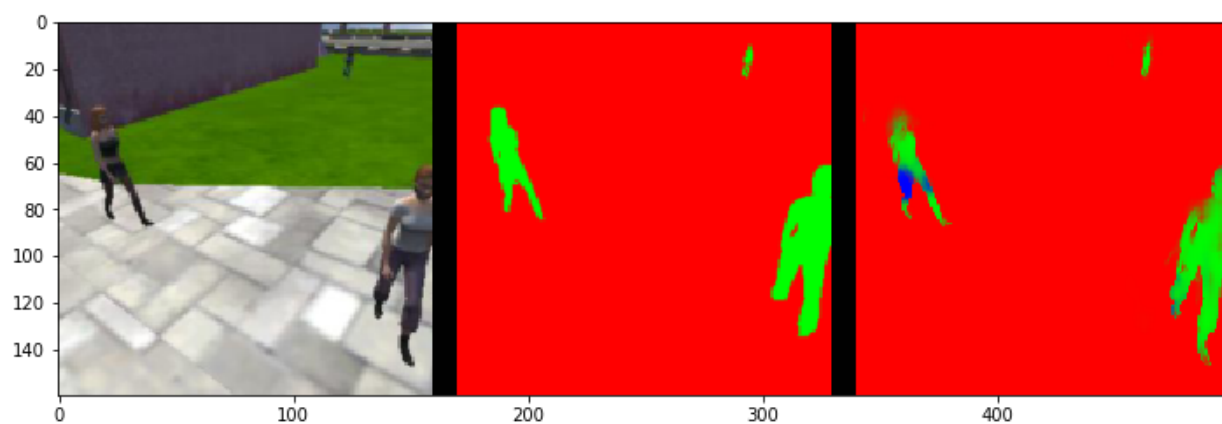


Image without target

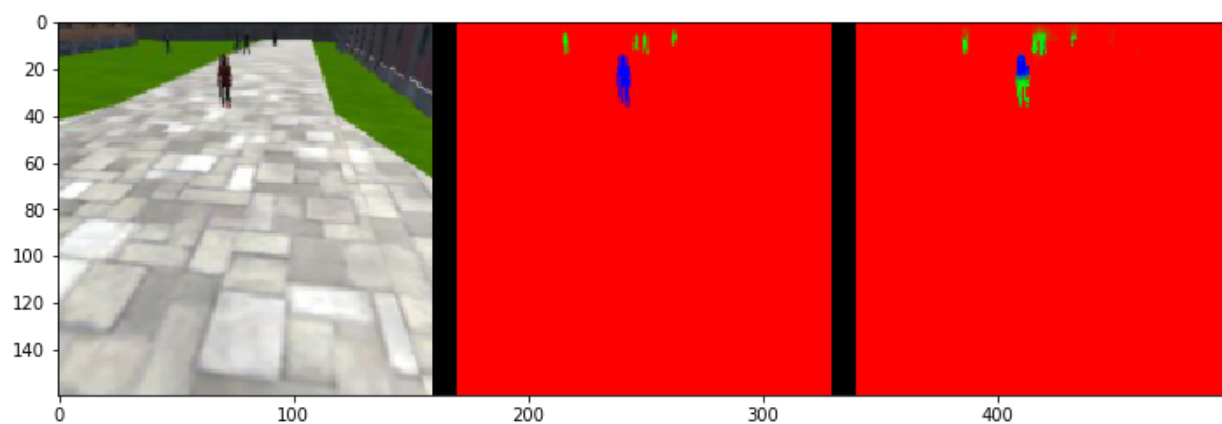


Image with target far away

Classification of other objects

My network identifies humans (green color), the target human (blue color) and everything else is red. Therefore at the moment it cannot identify dogs, cats, cars etc. In order to identify those, the model should be trained with additional images that include dogs, cats, cars or other objects and increase the number of classes from 3 to 4,5,6 etc. Probably the network in this case would be more complicated with more encoders in order to be able to learn the shapes of the other objects.

Future improvement

An improvement for this project would be to collect more data and train the network on them and maybe creating a model with more encoders and decoders with increased depth and decreasing the learning rate.